Heuristic for license-aware, performant and energy efficient deployment of multiple software in Cloud architecture

Eddy Caron, Arthur Chevalier, Noëlle Baillon-Bachoc, Anne-Lucie Vion

▶ **To cite this version:**

Eddy Caron, Arthur Chevalier, Noëlle Baillon-Bachoc, Anne-Lucie Vion. Heuristic for license-aware, performant and energy efficient deployment of multiple software in Cloud architecture. ICICS 2021 - 12th International Conference on Information and Communication Systems, May 2021, Valencia, Spain. 10.1109/ICICS52457.2021.9464578 . hal-03572922

HAL Id: hal-03572922
https://hal.science/hal-03572922

Submitted on 14 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Heuristic for license-aware, performant and energy efficient deployment of multiple software in Cloud architecture

Eddy CARON*, Arthur CHEVALIER*†
* *Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP*
F-69342, LYON Cedex 07, France
{firstname.lastname}@ens-lyon.fr

Noëlle BAILLON-BACHOC†, Anne-Lucie VION†
† *Orange S.A.*
noelle.baillon@orange.com
annelucie.cosse@orange.com

***Abstract*—In the Cloud Era, we want to be able to quickly deploy any software anywhere in the world to provide high availability and fast services while maintaining acceptable levels of performance, low energy consumption and ensuring the compliance with every software level agreements contracted. To answer some of these needs, different tools exist in parallel to a big variety of Cloud architectures. Several interesting problems arise like deployment, networking, storage, security, and many others. In this paper, we will focus on the deployment issue with a Software Asset Management point of view. Most Cloud providers use proprietary software to ensure different kinds of services, and with them comes the licensing problem. We will tackle and propose a heuristic to solve the problem of deploying software in a Cloud architecture while considering license compliance, license price, and other important criteria. We will prove the NP-completeness of this problem and compare our heuristic with others to evaluate the enhancement we propose.**

***Keywords*-Licensing, Cloud, Software Asset Management, Resources Management, Deployment**

I. Introduction

With the advent of Cloud technologies, many problems arise like networking, security, storage and more specifically in our case: deployment. Indeed, the problem of choosing the best place to deploy a virtual machine or a product directly is well known. The new kind of architecture and the need to add more variables (availability, performance, energy, etc.) to this problem makes it difficult. In our case, we describe a heuristic to answer a rising kind of problem and generally misleadingly understood: the software licensing issue. The software licenses are often perceived as one product equal to one key or license, which is wrong. The notion of software licenses is broader than that, especially with proprietary software. First, a license is only the representation of your right to use an instance of a product. The need to input a key inside a license verification process is just an editor way of checking if you possess it but for large infrastructure, it becomes very impractical. You can just possess the licenses and use the software without assigning them. This brings the notion of audit: the audit is the process of checking if you have enough licenses and is conducted by the editor or a third-party assigned by the editor. They will match your use of their software and your stock of licenses and you have to possess the exact amount or more licenses otherwise you will suffer penalties. These penalties can result in a huge amount of money as we saw with the AB InBev case [1] where SAP wanted US$600 million in damages for breaking the software license agreement. Secondly, the one-one relation between software and a license is wrong. Indeed, you can need a lot of licenses to possess the right to use only one instance of a product. This number of licenses is given by what we call a *metric* (in italic to not confuse with the more general metric term). This *metric* is a description of how to count the number of licenses depending on the usage of the software. These *metrics* can be in any format, generally described in the contract with the editor. It can go from a simple mathematical expression to full paragraphs of legal text explaining what you can or can't do. This method of licensing software is already time-consuming with standalone architectures and with the rise of virtualization, containerization, and Clouded architectures, it becomes really hard to ensure compliance at all times.

Considering only the problem of licensing for one deployment in a Cloud or not is not something to do. Indeed, besides the fact that this is not realistic, it would lead to terrible situations where your products could be deployed in bad places without availability, bad networking and performance. We also want to deploy multiple products at the same time to avoid getting local optimums. In this paper, we will focus on three parameters: the license cost, the performance, and the energy consumption.

In summary, this paper proposes a new heuristic for the deployment of several products and with a multi-parametric approach. We will start by stating the previous works in the different domains in Section II. We then prove that this problem of deployment is NP-Complete in Section III before describing the new heuristic in Section IV. Next, we evaluate our new heuristic and show the gain of using it in Section V before concluding in Section VI and presenting what can be enhanced in future works in Section VII.

II. State of the Art

One of the first publications about licensing was devoted to the study of automated *metrics* of software, the assessment of CIM repository and management of software assets [2]. Real Software Asset Management considerations started in 1999 when a study about the model and identification of software was proposed [3]. This study stated that SAM can mitigate technical, legal, managerial, financial and ethical risks in organizations. Later, in 2005, the need for a framework for control of software assets throughout their lifecycle to ensure long-term and efficient management has been show [4]. In 2011, a proposition to combine IT, processes and SAM was offered and revolved around four points:

- Being able to discover software
- Being able to make precise inventories of licenses and the infrastructure
- Implementing contract management
- Producing reports about readiness towards compliance and verification

Despite being a modern key to enable digitalization in companies, IT managers fail to address SAM issues and ignore the necessity of having a proper framework for compliance verification and vulnerability analysis [5]. Less than 20 percent of companies effectively use SAM [6] and while 65% of enterprises are audited yearly (up to 23% of them were audited three times or more on the same year), only 29% have an automatized monitoring of their systems and 25% of them have no monitoring at all. Such lack of SAM could result in huge expenses for companies [7] and can sometimes lead to severe trouble like the 2000 year problem that led to enormous costs to firms that didn't have SAM inventory databases [8]. While business, for security purposes, set up centralized security policy and denied to their employees the right to install software on the professional work station, the BYOD paradigm tends to reverse this situation and let people use as they see fit their devices leading to IT risks [9]. Academic field showed the heavy financial risks in case of SLA breach [10] and backed up the Flexera Study [11] by showing the lack of SAM in companies.

On the needs to give efficient tools to SAM processes, in 2017, a patent [12] proposed tools for discovering and collecting information on instances of software used in monitored environment. At the same time, the Cloud was added in SAM considerations in a review of existing SAM tools and a new SAM model for Cloud architectures [13]. Then, in 2018, a paper [14] proposed a new way of handling *metrics* in Cloud environments for products like Oracle Database and showed that an efficient deployment algorithm focused on software licenses could save money. The same year, Mann [15] proposed optimization of the placement of virtual machines with multiple parameters including license costs and showed that handling both problems of mapping virtual machines to physical machines and mapping applications to virtual machines leads to better results than considering the two problems in isolation. Even so, the problem is well formulated, it uses the fact that an application uses one license at most, as we see in his UML model, which is unrealistic but as we explain before this number of licenses rely on a variety of parameters like the underlying architecture. In summary, very few research have been made to enhance SAM processes or even to automate them.

On the contrary, lots of research has been done on the energy side of deployment in local and remote infrastructure [16]. Besides, the energy criteria become more and more important in today's world. Research has been conducted to reduce these consumptions and heuristics for multi-criteria deployments have been proposed such as *GreenPerf* [17] which introduces a performance and power consumption ratio to enhance energy efficiency. While there are numerous papers on this kind of multi-parametric deployment heuristic, none has been proposed taking into account Software Asset Management perspectives or with some confusions like shown before. Moreover, no work has been done on deploying multiple products on the Cloud while optimizing several parameters including SAM considerations. We will tackle this issue.

III. Proposed problem and proof of NP-completeness

In this section, we describe the problem and then prove that this problem is NP-complete. We want to deploy multiple products on a set of servers in the Cloud while optimizing three criteria: Energy consumption, Performance of products and License consumption. In other words, we want all of our products to consume the minimum energy and number of licenses while being given the maximum performance. Clearly, these objectives are contradictory, forming the basis for multi-parametric optimization. Our optimization problem comes from the fact that each time we deploy a product, the energy and performance of the selected server will change and the number of licenses can change anywhere depending on the *metric* so it is highly dynamic.

With the following variables:

- n Number of software to deploy.
- s Number of servers.
- α_i Cost of license for software i.
- β_j Cost of using one core on server j, energetically speaking.
- $m_{i,j}$ Number of licenses consumed for software i on server j. If software i is not on j then $m_{i,j} = 0$.
- r_i Resource of software i: here the number of cores required to install the software.
- R_j Resource of server j: here the number of cores of the server.

We define two problems:

Definition 1. *The optimization problem $OpTISAM$ is defined with the following equations:*

$$min \sum_{i=1}^{n} \left[\left(\sum_{j=1}^{s} m_{i,j} \right) \times \alpha_i \right] \tag{1}$$

$$min \sum_{server\ j\ used} R_j \times \beta_j \tag{2}$$

subject to:

$$\forall\ j \sum_{all\ deployments\ i\ on\ server\ j} r_i \ \leq \ R_j \tag{3}$$

which is a physical constraint where the cores used from deployment on a single server cannot exceed the number of cores of that server.

Definition 2. *The decision problem $SAMDec$ is defined as: given two constraints, B_E and B_L which are energy budget and license budget respectively, is there a deployment to fulfill the following equations:*

$$\sum_{i=1}^{n} \left[\left(\sum_{j=1}^{s} m_{i,j} \right) \times \alpha_i \right] \leq B_L \tag{4}$$

$$\sum_{server\ j\ used} R_j \times \beta_j \leq B_E \tag{5}$$

also subject to Equation.3.

Lemma. *$SAMDec$ is NP-complete.*

Proof:

Definition 3. *Let I_1 be an arbitrary instance of $2PARTITION - EQUAL$ with:*
$2n$ integers $a_1, ..., a_{2n} \leq 1$ where $\sum_{i=1}^{2n} a_i = 2S$

$$\exists\ I\ subset\ of\ \{1, ..., 2n\}\ /\ |I| = n \sum_{i\ \in\ I} a_i = S$$

Definition 4. *Let I_2 be an instance of $SAMDec$ with:*
n products to deploy with $r_i = 1 = m_{i,j}$ and $2n$ servers with $R_j = 1$ hence one product per server and $s = 2n$.
With $Used$ declared as the set of used servers (by indices), $n = |Used|$ iff there is a solution:
Let $\beta_j = a_j$:

$$\sum_{server\ j\ used} R_j \times \beta_j = \sum_{server\ j\ used} a_j = S$$

Let $\alpha_j = X - a_j$:

$$\sum_{server\ j\ used} \left[\left(\sum_{software\ i\ on\ j} m_{i,j} \right) \times \alpha_j \right] = \sum_{server\ j\ used} (X - a_j) = nX - S$$

Moreover, if I_2 has a solution with B_E and B_L fixed then we have a solution $Used$ for the following equations:

$$\sum_{j\ \in\ Used} \beta_j \ \leq \ B_E$$
$$\sum_{j\ \in\ Used} \alpha_j \ \leq \ B_L$$

With the following:

- $Size(I_1) = 2n + log\ \sum_{i=1}^{2n} a_i$
- $Size(I_2) = 2n + log\ \sum_{i=1}^{2n} \alpha_i + log\ \sum_{i=1}^{2n} \beta_i$

We can conclude that I_2 has a polynomial-size in I_1 and we prove that I_1 has a solution $\iff$ I_2 has a solution:

I_1 has a solution I
$|I| = n$ and $\sum_{i\ \in\ I} a_i = S$
We take $Used = I$ and have a solution to I_2 because:

$$\sum_{j\ \in\ I} \beta_j \times R_j = \sum_{j\ \in\ I} \beta_j = S$$
$$\sum_{j\ \in\ I} \left[\left(\sum_{metric\ i\ on\ j} m_{i,j} \right) \times \alpha_j \right] = nX - S$$

hence a solution to I_2.

I_2 has a solution $Used$
$|Used| = n$
We take I as $Used$ and have a solution to I_1 because:

$$\sum_{j\ \in\ Used} \beta_j = \sum_{j\ in\ Used} a_i = S$$
$$\sum_{j\ \in\ Used} \alpha_j = nX - \sum_{j\ \in\ Used} \beta_j = nX - S$$

altogether with $|Used| = n$, we have a solution to I_1.

■

Therefore, the problem of deploying one product per server and only optimizing energy and license price is proved NP-complete. As our problem add the performance criterion and allow multiple deployments of products onto the same server, it is a superclass of the $SAMDec$ problem so our problem is NP-complete too.

IV. New heuristic

To tackle this problem, we propose a model of the deployment and a heuristic using that model and giving good results as we will see in Section V. We chose to model the deployment with a tree so it handles all our constraints. Effectively, each layer of the tree will be a new product deployed and each node will be a server. The root node will be the starting point and going from one node to another will describe the deployment of the product to the server corresponding to this node. Each node has a set of attributes corresponding to the different criteria: energy, performance and license consumption. We handle the dynamicity of the criteria with the attributes of the nodes: When a node representing a server s have a node representing the same server then the attributes of the son will be modified as its energy E_s, its performance P_s and its license consumption L_s have evolved. Also, the license consumption of all servers possibly evolves with each deployment as *metrics* definitions are very sparse. You can clearly understand this behavior in Figure 1.

While the generation of this tree is easy to do, such a model of deployment takes a lot of space especially in Cloud architecture where there are thousands of available servers. We computed that we need tens of GB of memory for 4 products and 150 servers.

To keep acceptable results and as the problem is NP, we can't go through the entire tree to search for the optimal result. Therefore, we have to use a heuristic in an attempt to get as close as possible to the optimal. We will use the *GreenSAM* heuristic to select nodes to explore. This heuristic is based on a scoring function taking the three criteria:

$$Score = \frac{P_s/M_P}{E_s/M_E + L_s/M_L + 1} \tag{6}$$

with the following variables:

- E_s Energy of server s
- L_s License consumption of server s
- M_E Maximum of energy over the server subset
- M_L Maximum of license consumption over the server subset
- M_P Maximum of performance over the servers subset
- P_s Performance of server s

If one of the maximum is zero, we ignore the part used by it so if M_E is equal zero then E_s/M_E is reduced to zero.

We then explore all nodes, one by one, following the resulting order. Each time we explore a node we use the heuristic with the available sons until reaching a leaf. To compute the overall score of the deployment when reaching the leaf, we sum the score of each node in the path. If the score is better than the global score found until here, we save it as the best result. Besides, as the memory is limited, each time we explore a node we have to generate the according sons with their attributes. As we want to avoid putting all the products on the same server, we put a zero performance index to servers already full of products (a server is full when the sum of the cores used by the products installed on it equal the server cores). All the licenses' consumptions are computed as well as the energy. As the computation of the energy consumed by a product is a difficult thing to do, we chose to compute energy with the usage of a server or not; i.e. if a server is used by one of our product then we add its energy. When we deploy one product on a server already using one of our product, then we will not consume anymore energy.

V. Evaluation

With the sheer number of 2,227.33 PB of memory for 50 servers and 10 *metrics*, it becomes obvious that we cannot go through the entire tree. Therefore, a good heuristic is required. To evaluate the different heuristics, we impose a memory limit (directly related to the number of nodes to go through) and see how they compare one to each other's before comparing them to the optimal solution when we can compute it.

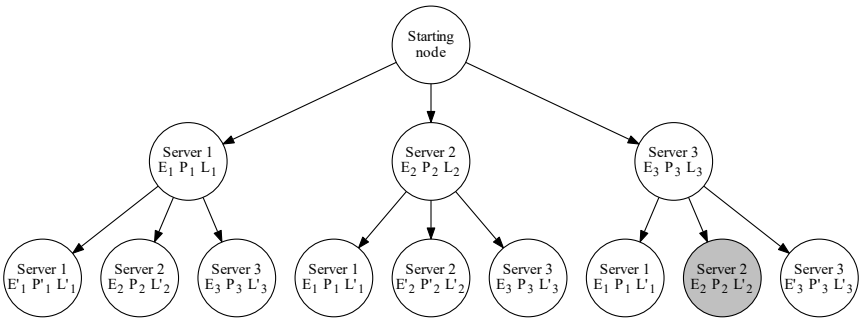

Figure 1: Deployment of two products over three servers. We can see here that if we arrive at the grey node it will mean that we deployed product 1 on server 3 and product 2 on server 2, therefore, our final scores will be $E_T = E_3 + E_2$, $P_T = P_3 + P_2$, $L_T = L_3 + L'_2$. Our goal is to minimize E_T and L_T while maximizing P_T.

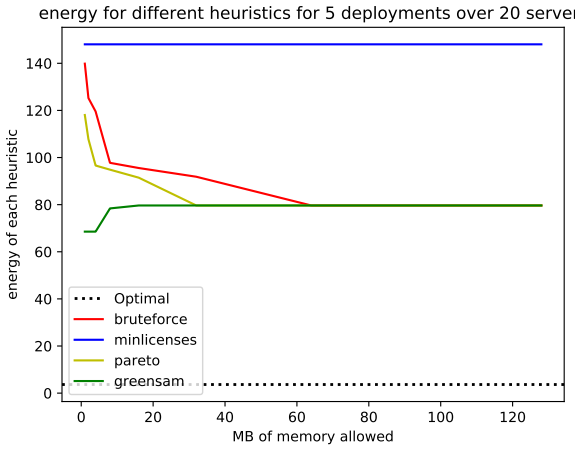

Figure 2: Energy consumption criterion

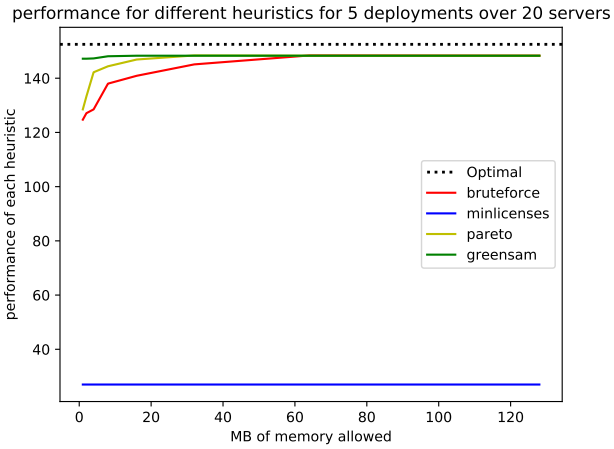

Figure 4: Performance criterion

We can compare heuristics with the $Score$ function. Indeed, Equation.6 returns a result in the range $[0; 1]$ for each node. We can deduce that if the score is 1 then we have the most performant server and that both energy and license consumptions are 0 which is the best score. When the result tends to 0, it means that the performance is terrible or that the energy or license tends to the maximum, which is not a suitable solution. The overall score is the sum of the scores of each node in the path to the leaf. Therefore with n products, the overall score is in $[0; n]$. Most of the time the optimal solution will be inferior to n as we don't have, in real-world situation, servers that don't consume anything for each deployment. Therefore, we can't assume that the optimal is n for the following evaluations.

We compared the following heuristics to see how they behave in different situations: ***Bruteforce*** Enumerate the most possible nodes with Left-to-Right order. ***MinLicense*** Choose the best node to go through depending on the number of licenses it will consume. This heuristic is used to see how a single criterion optimization will behave. $Pareto$ This heuristic uses a Pareto front on the available nodes to go through. The nodes in the Pareto frontier will be searched from left to right and those not on the frontier will be ignored. ***GreenSAM*** Go through all nodes by sorting them with the *GreenSAM* function.

We evaluated our deployment algorithm with four different scenarios: two sizes of sets by two types of limit.

A. *Small deployment with memory limit*

We compared the four heuristics on a deployment of 5 products over 20 servers with the memory limit ranging from 1MB to 128MB. We can see the results for energy in Figure 2, license in Figure 3, performance in Figure 4 and score in Figure 5.

We can see that overall, the *GreenSAM* heuristic gets the best results on all criteria except in licenses where obviously the $MinLicenses$ algorithm is better. While the other algorithms take times to stabilize, *GreenSAM* find

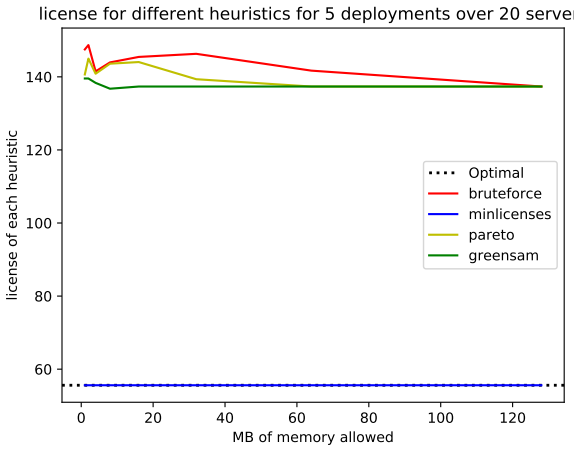

Figure 3: License consumption criterion

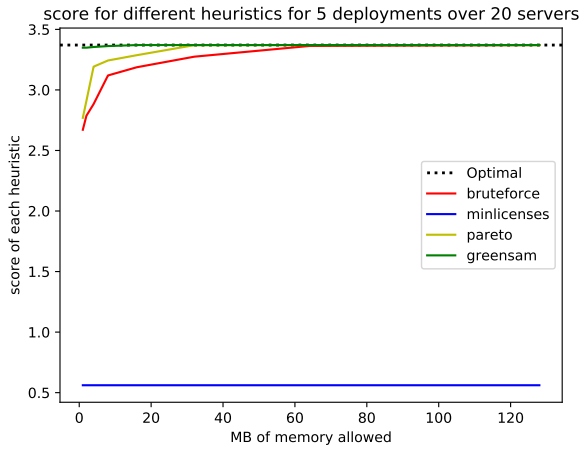

Figure 5: Score criterion

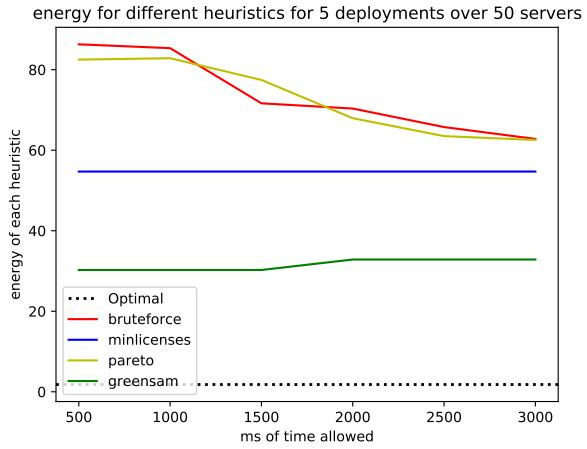

Figure 6: Energy consumption criterion

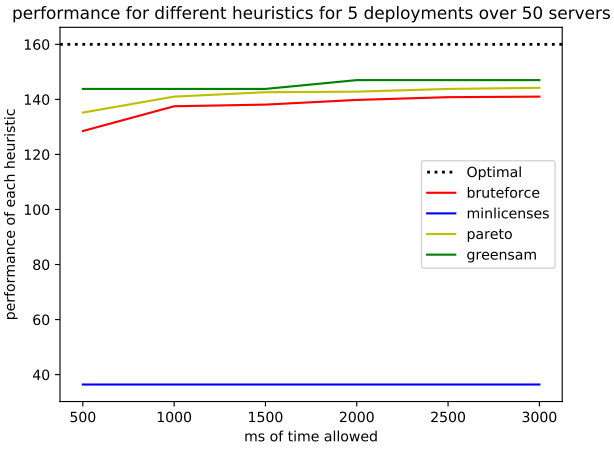

Figure 8: Performance criterion

quickly its best solution and is very close to the optimal performance.

B. Small deployment with time limit

For this evaluation, we used the same data set as the first evaluation but with a time limit ranging from 500ms to 3000ms. We can see the results for energy in Figure 6, license in Figure 7, performance in Figure 8 and score in Figure 9.

We see that *GreenSAM* beats the other algorithms in this configuration too. It stabilizes less quickly than with memory limit but manages to get a very good score, performance, and energy while sacrificing few licenses.

C. Huge deployment with memory limit

The third experiment generated deployments of 50 products over 1000 servers and compared the results of the heuristics with memory limits ranging from 1MB to about 300MB. We changed the scale to a logarithmic one for clarity. We can see the results in Figure 10.

With the small dataset, all heuristics tend to converge to the optimal. We can see here that there is a big gap in the score criterion. The final score of the *GreenSAM* heuristic was 43.23 while the other algorithms got 2.6 for $Bruteforce$, 0.13 for $MinLicenses$ and 10.7 for $Pareto$. With just 1MB of memory, meaning going through 57,000 nodes over 50^{1000}, *GreenSAM* finds an excellent solution that has the best performance overall and very few licenses at, unfortunately, the cost of energy.

D. Huge deployment with time limit

On this last evaluation, we reused the data set from the third evaluation and set time limit ranging from 500ms to 3000ms like the second evaluation. We can see the results from this experiment in Figure 11.

We can see that under 1.5 seconds the $Pareto$ algorithm don't manage to reach a single leaf because of the amount

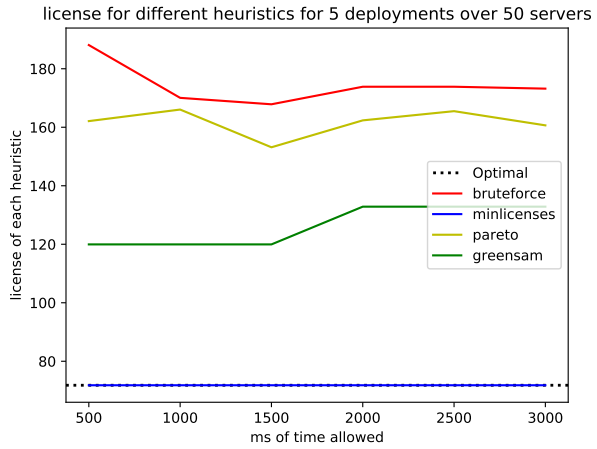

Figure 7: License consumption criterion

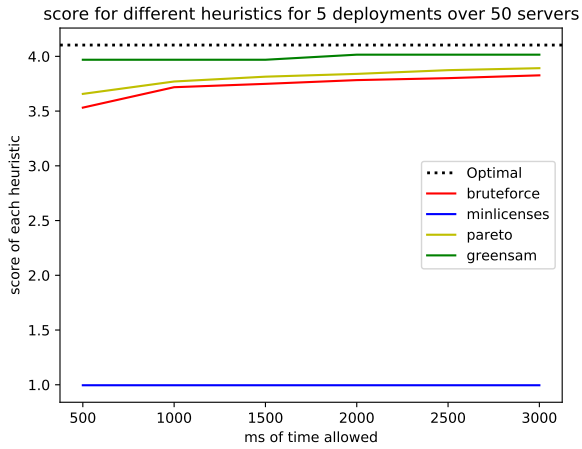

Figure 9: Score criterion

Figure 10: Criteria results for large deployment

of computation to do but once it does, it gets good results. The *GreenSAM* heuristic is also the best overall in this configuration and reaches the score of 43.17 with only 500 milliseconds.

From these four scenarios, we can conclude that with both memory limit and time limit, the *GreenSAM* heuristic is better than the 3 others. Besides, it achieves to get good results overall with very limited resources (1MB for memory and 500ms for time limit). Overall, the *GreenSAM* manages to obtain great performance compared to other heursitcs while sacrificing few licenses and energy. Finally, the *GreenSAM* heuristic is near the optimal on small sets and gets a very good score on the big sets.

VI. Conclusion

In conclusion, this paper introduces several new advances. First, it presents a deployment problem with Software Asset Management considerations like [15] but with revisited license consumption computation to adjust it to real-world usage. Then, we proposed representation and a structure for this deployment problem in a tree-shaped manner. We gave a proof of NP-completeness of the decision problem representing our deployment. We then introduced the main contribution: a heuristic that optimizes several criteria and manages to get good results with the deployment of one product on the Cloud compared to other heuristics. We can see from the evaluation that the *GreenSAM* heuristic needs a very little amount of memory to obtain near-optimal results and that it does it quickly. Besides, compared to other heuristics, *GreenSAM* ensures compliance at deployment time by removing servers that will put us in a non-compliant state.

VII. Future works

Several works can be done to enhance the *GreenSAM* heuristics. First, we can think of a way to reduce the tree structure size by finding some subtree replicas and store them and their results to speed up the heuristic. Then, we could think of using a Pareto Front in *GreenSAM* to pre-select good subnodes and ignoring the ones that will surely not give good results. The problem of this latter method is that it is based on a single deployment and maybe an ignored node would have got good results in its descendants. It should also be interesting to add *metric* aware optimization at single product deployment level. We could categorize *metrics* and search optimization for each of them to enhance the license score function. Finally, a better $Score$ function could be used to sort the nodes in a better way or even a $Score$ function that takes into account several layers of the tree to get more accurate results.

References

[1] P. Sayer, "SAP settles licensing dispute with AB InBev." [Online]. Available: https://www.itworld.com/article/3264435/sap-settles-licensing-dispute-with-ab-inbev.html

[2] R. D. Banker and R. J. Kauffman, "Automated software metrics, repository evaluation and software asset management: New tools and perspectives for managing integrated computer aided software engineering (i-case)," *Information Systems Working Papers Series, Vol*, 1991.

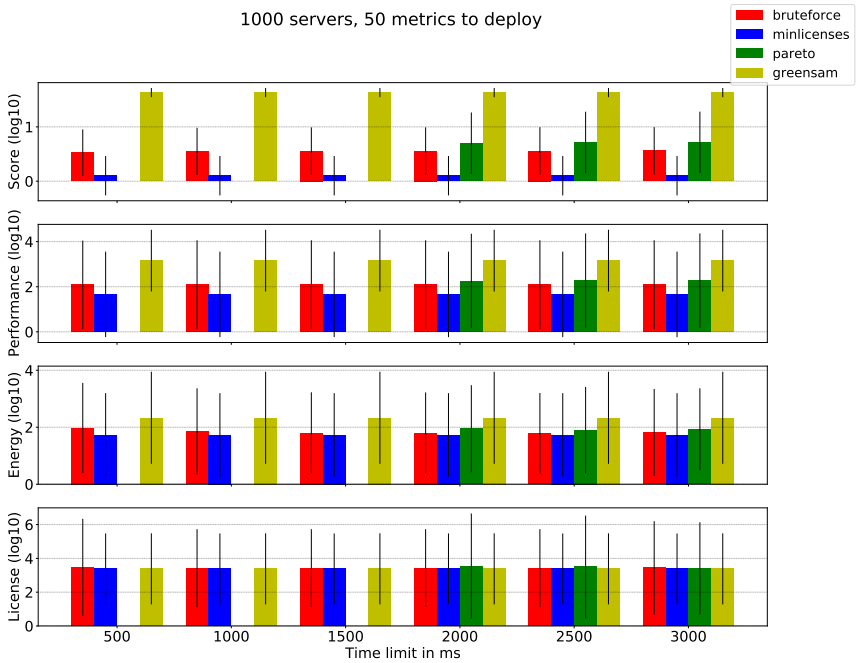

Figure 11: Criteria results for large deployment

[3] N. F. Holsing and D. Yen, "Software asset management," *Information Resources Management Journal*, vol. 12, no. 3, pp. 14–26, Jul. 1999.

[4] M. Ben-Menachem and G. Marliss, "IT assets—control by importance and exception: Supporting the "paradigm of change"," *IEEE Software*, vol. 22, no. 4, pp. 94–102, Jul. 2005.

[5] K. Dempsey, N. Goren, P. Eavy, and G. Moore, "Automation support for security control assessments: Software asset management," National Institute of Standards and Technology, Tech. Rep., 2018.

[6] J. E. Mbowe, I. Zlotnikova, S. S. Msanjila, and G. S. Oreku, "A conceptual framework for threat assessment based on organization's information security policy," *Journal of Information Security*, vol. 05, no. 04, pp. 166–177, 2014.

[7] A. M. Q. Varela, M. P. Méxas, and G. M. Drumond, "The scenario of software asset management (SAM) in large and midsize companies," *Independent Journal of Management & Production*, vol. 9, no. 2, p. 301, Jun. 2018.

[8] M. Ben-Menachem, "Towards management of software as assets: A literature review with additional sources," *Information and Software Technology*, vol. 50, no. 4, pp. 241–258, Mar. 2008.

[9] J. Swartz and P. Vysniauskas, "Software asset management in large scale organizations- exploring the challenges and benefits," Master's thesis, University of Gothenburg, 3 2015.

[10] F. Dzerzhinskiy, "About lawyers, programmers, and software assets," Mar. 2012.

[11] Flexera, "How security risks & the shift to the cloud are transforming sam," Tech. Rep. [Online]. Available: https://resources.flexera.com/web/pdf/WhitePaper-SLO-Security-Risks-Cloud-Transforming-SAM.pdf

[12] P. Gocek, P. Kania, B. Malecki, M. Paluch, and T. Stopa, "Obtaining software asset insight by analyzing collected metrics using analytic services," uS Patent 9,652,812.

[13] N. Baillon, A.-L. Vion, N. D. Palma, and F. Boyer, "Software license optimization and Cloud computing," *CLOUD COMPUTING 2017*, p. 125, 2017.

[14] N. Baillon, E. Caron, A. Chevalier, and A.-L. Vion, "Towards economic and compliant deployment of licenses in a Cloud architecture," San Francisco, USA, Jul. 2018, hal-01808751. [Online]. Available: https://hal.inria.fr/hal-01808751

[15] Z. A. Mann, "Resource optimization across the cloud stack," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 1, pp. 169–182, Jan. 2018.

[16] I. Foster and C. Kesselman, "Computational grids," in *Vector and Parallel Processing — VECPAR 2000*, J. M. L. M. Palma, J. Dongarra, and V. Hernández, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 3–37.

[17] D. Balouek-Thomert, E. Caron, and L. Lefevre, "Energy-aware server provisioning by introducing middleware-level dynamic green scheduling," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. IEEE, May 2015.